MULTIMEDIA UNIVERSITY

STUDENT ID NO

MULTIMEDIA UNIVERSITY

FINAL EXAMINATION

TRIMESTER 1, 2017/2018

ECE4246 – ADVANCED OBJECT-ORIENTED DESIGN WITH JAVA

(All Sections / Groups)

14 OCTOBER 2017
2:30 P.M – 4:30 P.M.
(2 Hours)

INSTRUCTIONS TO STUDENT

1. This examination paper consists of SEVEN (7) pages, including the cover page, with THREE (3) questions only.
2. Answer ALL questions. All questions carry 25 marks and the distribution of the marks for each question is given.
3. This examination is OPEN-BOOK. Allowed material: Textbook and/or bound notes only.
4. Please print all your answers in the Answer Booklet provided.

Question 1

Create a game where user is asked to predict the values from rolling a regular six-sided die.

(a) Write a class, **`OutsideBoundsException`**, which inherits from class **`Exception`** (from *java.lang*).

[1 mark]

(b) Write a class, **`Die`**, which simulates the rolling of a 6-sided die. Provide two static integer constant instance variables, **`MIN`**=1, and **`MAX`**=6, to represent the minimum and maximum values of a die roll. Also provide the following static methods:

- **`roll()`**, which generates and returns a random integer between **`MIN`** and **`MAX`**, inclusive.
- **`checkValidity(int)`**, which throws an **`OutsideBoundsException`** whenever the argument is less than **`MIN`** or greater than **`MAX`**. Provide meaningful exception messages to indicate whether the value is below or above the bounds. The method does not have a return value.

[8 marks]

(c) Write an application class, **`NumberGame`**, which allows the user to repeatedly predict the outcome of a die roll, until the user enters "q" (meaning, quit). For every iteration, prompt the user to enter a value (the prediction) and then simulate the roll of a die using the method **`roll`** from class **`Die`**. Display the generated roll value, compare it with the user-predicted one, and if they match, display the message "You predicted correctly!"

[12 marks]

Catch and handle all exceptions that may occur.

[4 marks]

```
Give me a prediction ('q' to quit):
4
Computer rolled 6
Give me a prediction ('q' to quit):
8
Value above upper bound.
Give me a prediction ('q' to quit):
0
Value below lower bound.
Give me a prediction ('q' to quit):
2
Computer rolled 2
You predicted correctly!
Give me a prediction ('q' to quit):
q
```

Figure Q1: Sample run of the program

Continued...

Question 2

Create a program which lists all the courses (or subjects) in an undergraduate program.

(a) Write a class, **`Course`** (see Figure Q2). Instance variable **`title`** is the title of the course (eg., "Java Programming") and **`code`** is a 4-digit number, stored as a **`String`** (eg., "4246"). Provide a constructor, mutator and accessor methods, and a **`toString`** method, as shown in Figure Q2. Mutator method **`setCode`** must ensure that the argument is a 4-digit **`String`** value before setting **`code`** to it. For arguments that do not fulfill the condition, set the code to default "0000".

[10 marks]

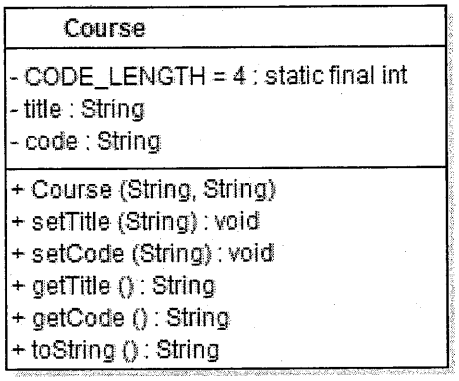
Course
- CODE_LENGTH = 4 : static final int
- title : String
- code : String
+ Course (String, String)
+ setTitle (String) : void
+ setCode (String) : void
+ getTitle () : String
+ getCode () : String
+ toString () : String

Figure Q2: Class Diagram of **`Course`**

(b) Write a **`Comparator`** (from *java.util*) class for **`Course`** objects, **`CourseTitleComparator`**, compares the title attribute.

[4 marks]

(c) Write a class, **`UGProgram`**, which has two instance variables. They are a **`String`**, **`name`** (eg., "BSc Bluffology"), and an **`ArrayList`** of **`Course`** objects, **`courses`**. Provide a constructor, and mutator and accessor methods for **`name`**. Also provide the following methods

- **`addCourse`**, which adds a course to **`ArrayList`** courses.
- **`listCoursesByTitle`**, which returns a **`String`** of the list of all courses in the undergraduate program, sorted by the course titles.
- **`toString`**, which returns the name of the undergraduate program and the number of courses it contains.

[11 marks]

NOTE: DO NOT write the driver program.

Continued...

Question 3

Create a program to compute bonus for employees based on their performance. The company uses 3 rating levels: 1, 2, and 3. Figure Q3 shows the code for **Employee**.

```
public abstract class Employee
{     private String name;
      private double salary;

      public Employee (String nm, double sl)
      {     setName(nm);
            setSalary(sl);
      }

      public void setName (String nm) {    name = nm;   }

      public String getName () {     return name;        }

      public void setSalary (double sl) { salary = sl; }

      public double getSalary () {  return salary;      }

      public String toString ()
      {     return name + ", $" + salary;         }

      public double getBonus ()
      {     double bonus;
            int rating = performanceRating();

            switch (rating)
            {     case 2:
                        bonus = 1.5 * salary;
                        break;
                  case 3:
                        bonus = 2.5 * salary;
                        break;
                  default:
                        bonus = 0;
            }
            return bonus;
      }

      public abstract int performanceRating (); //levels 1-3 only
}
```

Figure Q3

(a) Write a class, **SalesPerson**, which inherits from **Employee**. The class has an attribute, **sales**, which is type double and represents the amount of sales the person has made. Provide a constructor. Define the inherited method, **performanceRating**. The performance of a salesperson is determined by the amount of sales. Table Q3a shows the ratings.

Continued...

Sales Amount	Performance Rating
Less than $20,000	1
$20,000 - $49,999	2
$50,000 or more	3

Table Q3a

[5 marks]

(b) Write a class, **`ProjectManager`**, which inherits from **`Employee`**. The class has an attribute, **`onTimeCompletionRate`**, which is a value between 0.0 and 1.0 (inclusive) and represents the rate of projects that are completed on time. Provide a constructor. Define the inherited method, **`performanceRating`**. The performance of a project manager is determined by the project on-time completion rate. Table Q3b shows the ratings.

Project On-Time Completion Rate	Performance Rating
Less than 0.7	1
0.7 or more, less than 0.9	2
0.9 or more	3

Table Q3b

[5 marks]

(c) Write a **`JPanel`** class, **`PerformanceReview`**, as shown in Figure Q3c-i.

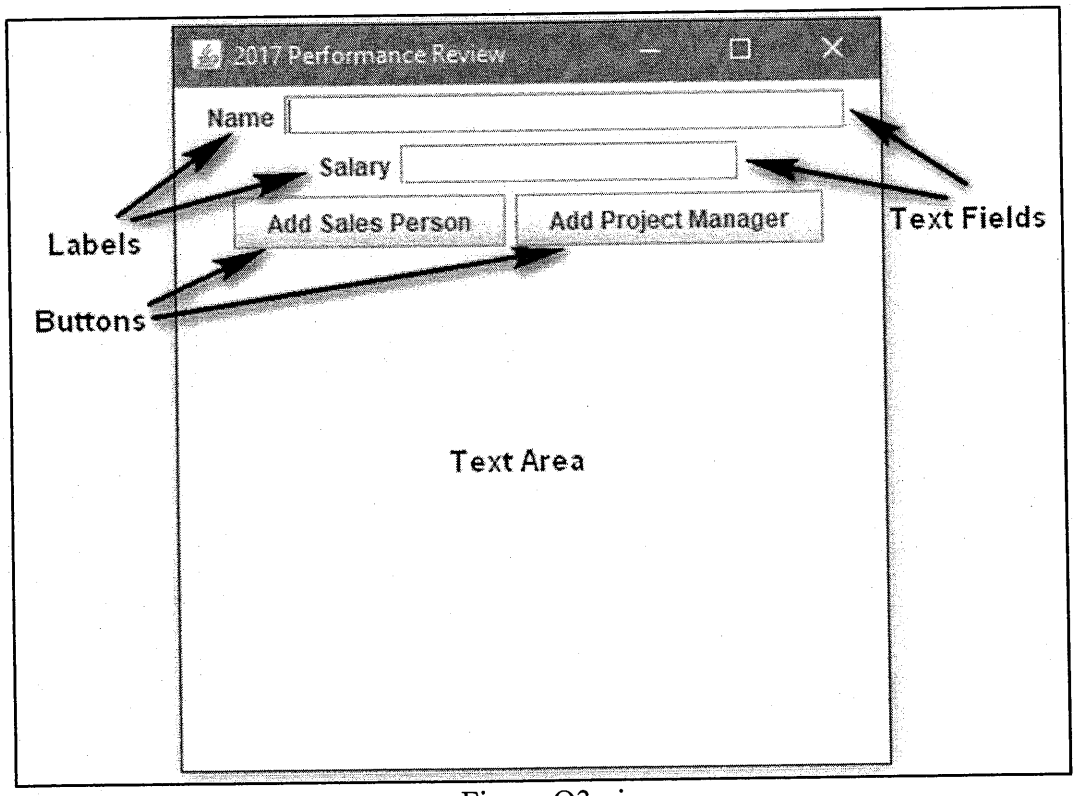

Figure Q3c-i

Continued...

In addition to the GUI elements, create an **`ArrayList`** of **`Employee`** objects to store each **`SalesPerson`** or **`ProjectManager`** object added by the user.

The user must key in the name and salary of the employee, then click either "Add Sales Person" or "Add Project Manager". Once a button clicked, show an input dialog box to allow the user to provide further information about the employee, either sales amount or on-time completion rate. See Figure Q3c-ii and Figure Q3c-iii.

Display each of the employee objects, including the bonus amount, added by the user in the text area. See Figure Q3c-iv.

[13 marks]

NOTE: DO NOT write the driver program.

(d) State ONE (1) design pattern used in the design of classes **`Employee`**, **`SalesPerson`**, and **`ProjectManager`**.

[2 marks]

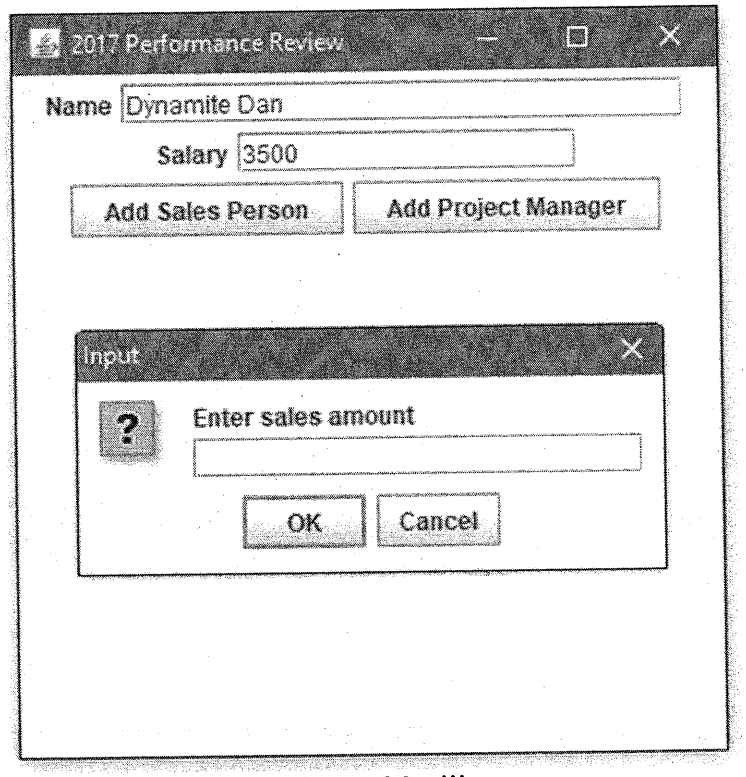

Figure Q3c-iii

Continued...

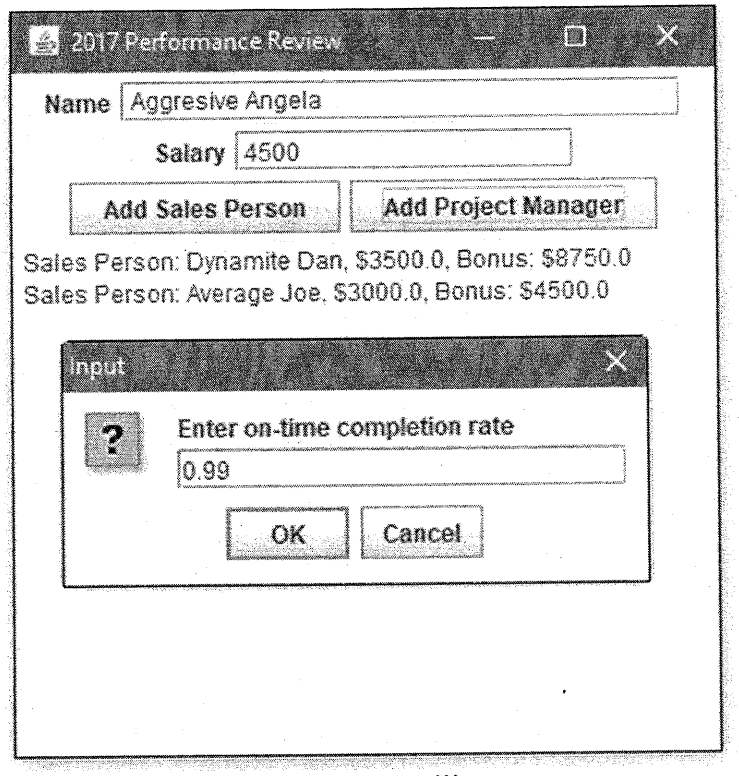

Figure Q3c-iii

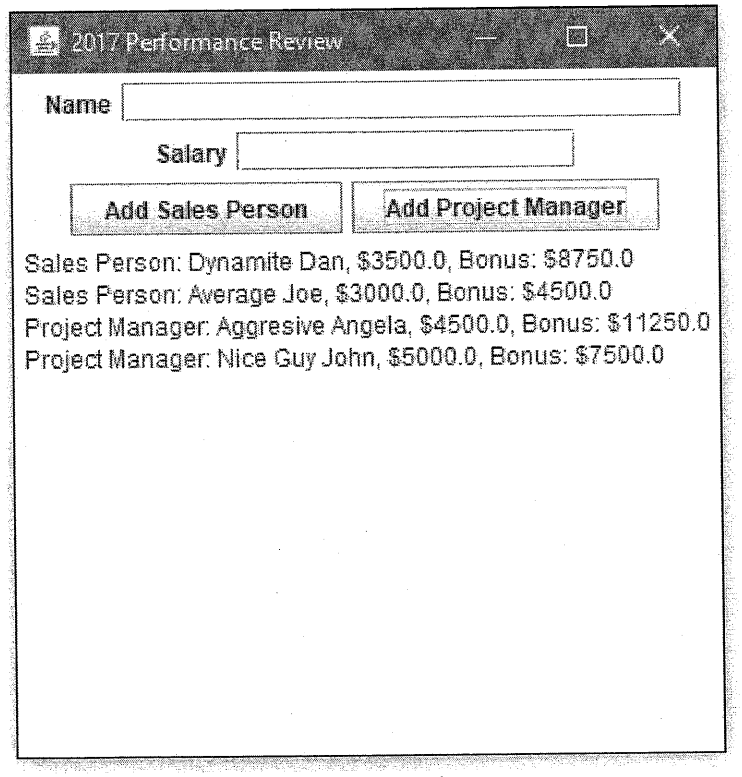
2017 Performance Review
Name
Salary
Add Sales Person
Add Project Manager
Sales Person: Dynamite Dan, $3500.0, Bonus: $8750.0
Sales Person: Average Joe, $3000.0, Bonus: $4500.0
Project Manager: Aggresive Angela, $4500.0, Bonus: $11250.0
Project Manager: Nice Guy John, $5000.0, Bonus: $7500.0

Figure Q3c-iv

End of Paper